Homework 2 Matlab Portion

This is the code and solutions for problem 5 of homework 2.

Code:

The following shows the MATLAB code that was generated to produce the relevant outputs for each part of the question. Each part of the code is labeled to denote which parts of the code are used for each section.

```
%%ECEM146: Homework 2
% Author: Thomas Kost
% UID: 504989794
% Date: 4/9/20

%% Problem 5:
clc,clear; close all;
%% 5a

training = dlmread('regression_train.csv');
data_scatter = figure(1);
scatter(training(:,1), training(:,2));
xlabel('x');
ylabel('y');
saveas(data_scatter, '5a.jpg');
%% 5b
output_file = fopen('matlab_output.txt', 'w');
%Linear Regression

X = [ones(length(training),1),training(:,1)];
y = training(:,2);

w = (X'*X)\X'*y;
J = norm(X*w-y)^2;
fprintf(output_file, "For w0= %f, w1 = %f , we had J(w) = %f \n",w(1),w(2), J);

data_regression = figure(2);
hold on;
scatter(training(:,1), training(:,2));
f = @(x0,x1) w(1) +w(2)*x0-x1;
fcontour(f,[-0.5,1,-0.5,2.5],'--r', 'LevelList', 0)
xlabel('x');
ylabel('y');
hold off;
saveas(data_regression, '5b.jpg');

%% 5c
n =10000;
eta = [0.05, 0.001, 0.0001, 0.00001];
w = zeros(2,1);

for i  =1:length(eta)
```

```
    J_old= norm(X*w-y)^2;
    J_new=0; %#ok<NASGU>
    itterations =0;
    for j = i:n
        %update w
        z=((X*w-y).*X);
        grad = [sum(z(:,1));sum(z(:,2))];

        w = w - eta(i)*grad;
        J_new = norm(X*w-y)^2;
        if ((abs(J_old-J_new)) <0.0001)
            itterations=j;
            break;

        end
        J_old=J_new;
    end

    fprintf(output_file, "For eta = %f : %i itterations, J = %f \n",eta(i),itterations,J_old);
    fprintf(output_file, "For eta = %f : w0= %f, w1 = %f \n",eta(i),w(1),w(2));

end
%% 5D

%repeat previous algorithm
eta = 0.05;
w= zeros(2,1);
J_old= norm(X*w-y)^2;
J_new=0;
itterations =0;
n=40;
fprintf("\n");
grad_decent_progression =figure(4)
hold on;
scatter(training(:,1), training(:,2));
for j = 1:n+1

    if(itterations == 0 || itterations == 10 || itterations == 20 ...
        ||itterations == 30||itterations == 40)
        fprintf(output_file, "For eta = %f : %i itterations, J = %f \n",eta,itterations,J_old);
        fprintf(output_file, "For eta = %f : w0= %f, w1 = %f \n",eta,w(1),w(2));
        f = @(x0,x1) w(1) +w(2)*x0-x1;
        if(itterations == 0)
            fcontour(f,[-0.5,1,-0.5,2.5], '--r', 'LevelList', 0);
        elseif(itterations == 10)
            fcontour(f,[-0.5,1,-0.5,2.5], '--b', 'LevelList', 0);
        elseif(itterations == 20)
            fcontour(f,[-0.5,1,-0.5,2.5], '--c', 'LevelList', 0);
        elseif(itterations == 30)
            fcontour(f,[-0.5,1,-0.5,2.5], '--g', 'LevelList', 0);
        elseif(itterations == 40)
            fcontour(f,[-0.5,1,-0.5,2.5], '--k', 'LevelList', 0);
            end
    end
    %update w
    z=((X*w-y).*X);
    grad = [sum(z(:,1));sum(z(:,2))];

    w = w - eta*grad;
    J_new = norm(X*w-y)^2;
    if ((abs(J_old-J_new)) <0.0001)
        itterations=j;
        break;
    end

    J_old=J_new;


    itterations = itterations +1;
end
legend('Training data','0 itterations', '10 itterations', '20 itterations', '30 itterations','40 ittera
xlabel('x');
ylabel('y');
hold off;
saveas(grad_decent_progression, '5d.jpg');
fclose(output_file);

%% 5e
test = dlmread('regression_test.csv');
m = [0,1,2,3,4,5,6,7,8,9,10];
Erms = zeros(2,length(m));
for i =1:length(m)
    %generate phi
    phi_training=zeros(length(training),m(i)+1);
    %w= zeroes(m(i)+1,1);
    for x =0:m(i)
        phi_training(:,x+1) = training(:,1).^x;
    end

    w_training = (phi_training'*phi_training)\phi_training'*y;
    Erms(1,i) = sqrt(norm(phi_training*w_training-y)^2/length(training));


    phi_test = zeros(length(test),m(i)+1);
    for x =0:m(i)
        phi_test(:,x+1) = test(:,1).^x;
    end
    Erms(2,i) = sqrt(norm(phi_test*w_training-test(:,2))^2/length(test));

end

model_complexity = figure(3);
hold on;
plot(m,Erms(1,:));
plot(m,Erms(2,:));
legend("training", "testing");
xlabel("Model Complexity (Degree)");
ylabel("Root-Mean-Square-Error");
hold off;
saveas(model_complexity, '5e.jpg');
```

Results:

This section will be broken down by each part.

5a

We plotted the inputs against the outputs–this is shown in Figure 1. In visualizing the training data we can see that the data appears to follow at least a third order system. The data seems to follow a sine wave like pattern. We can guess that a linear regression will not be very effective in predicting the data. This is because the data doesn't follow a linear relationship. As a result, a linear regression will always produce a large error, and cannot correctly predict all regions of the data.

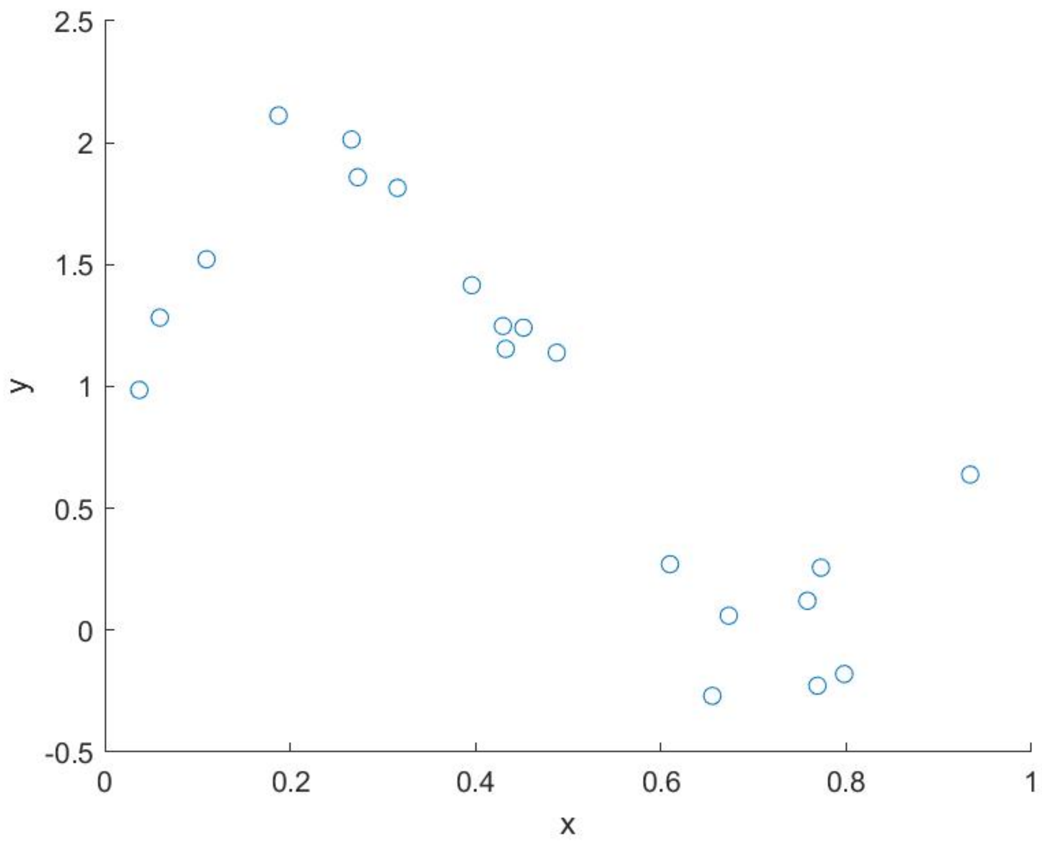

Figure 1: Input v.s. Output

5b

Using the results we generated in class we can find w using the closed form solution. Doing so we can generate the value of our linear regression and the associated value of our error function $J(w)$. We found $J(w) = 4.603635$ and

$$w = \begin{bmatrix} 1.991962 \\ -2.270484 \end{bmatrix}$$

These results were produced form the training data. The linear regression can be plotted agians out training data, so we can see the fit of the data to our regression line. This is shown in Figure 2. As we can see, the linear approximation does not fit the data well. While it does minimize the error, it does not fit the sinusoidal trend of the data.

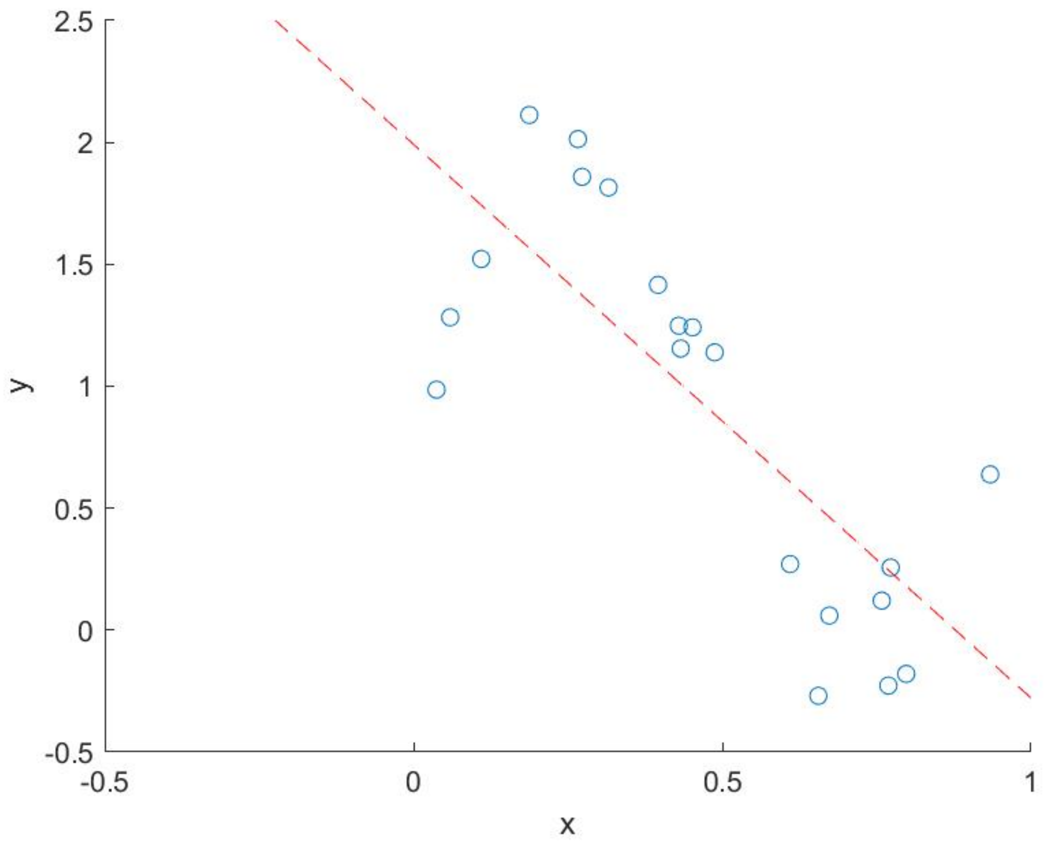

Figure 2: Linear Regression

5c

We implemneted gradient descent for different values of η. This produced the follwing result:

For eta = 0.050000 : 83 itterations, J = 4.604498

For eta = 0.050000 : w0= 1.980168, w1 = -2.246815

For eta = 0.001000 : 2 itterations, J = 4.604406

For eta = 0.001000 : w0= 1.980181, w1 = -2.246841

For eta = 0.000100 : 3 itterations, J = 4.604404

For eta = 0.000100 : w0= 1.980182, w1 = -2.246843

For eta = 0.000010 : 4 itterations, J = 4.604404

For eta = 0.000010 : w0= 1.980182, w1 = -2.246844

We can see that while the different learening rates did not prevent any of the executions of the algorithm from converging. We can see though that larger values of η the algorithm took more itterations to converge to a final value. This is likely due to the the algorithm taking too large of "steps", and as a result the algorithm was likely bouncing around the minimum. Regardless of the η used, we still converged to approximately the same value of w for each trial of the algorithm. We can compare this result of the algorithm to the closed form solution calculated in 5b. We can see that each run of the algorithm produced approximately the same value showing that the used learning rated did not affect the accuracy of the algorithm. However, we can clearly see that suboptimal learning rates caused the algorithm to require more itterations to converge to the same value. Our results suggest that step sizes below 0.001 are too small, as they are similar in order of magnitde of itterations. 0.05 is clearly too large and likely lead to some oscillation as described earlier.

5D

Here we implemnted the previous algorithm with only a learning rate of 0.05. This resulted in the following results at 0, 10, 20, 30, and 40 itterations.

For eta = 0.050000 : 0 itterations, J = 28.655524, w0= 0.000000, w1 = 0.000000

For eta = 0.050000 : 10 itterations, J = 7.629320, w0= 1.253005, w1 = -0.787455

For eta = 0.050000 : 20 itterations, J = 5.577512, w0= 1.572725, w1 = -1.429108

For eta = 0.050000 : 30 itterations, J = 4.917096, w0= 1.754114, w1 = -1.793141

$textForeta = 0.050000 : 40itterations, J = 4.704529, w0 = 1.857022, w1 = -1.999670$

We can ovserve that as the number of itterations increases the error J decreases.We can also see the values of each index in w gradually approaching the final values which we know the gradient decent will converge to (form 5c). We can also see that the changes become more gradual as the algorithm becomes closer to the final value that it converges to. This makes sense as the algorithm should be continually approaching the point minimal for our cost function. These results can be seen in the plot in Figure 3.

5E

We calculated the root mean square error for both the training and testing data when using the generalization of regression to polynomials of a higher degree. In doing so we were able to plot the error against model complexity–the degree of the system we were approximating with. This can be seen in Figure 4. This plot shows that for our training data root-mean-square error decreases with model complexity monotonically. However, this is not the case for our training data. We can see from our training data that our root-mean-square error is smallest between the 3rd degreee and 6th degree models of our system. Looking at the data itself we can clearly see that the system is at least a cubic function, and so this result makes sense. Looking at the root-mean-square errors of our testing data we can see that the smallest root-mean-square error is at our 6th degree model (error of 0.1705). As a result a 6th degree polynomial best fits our data.

We can also see evidence of both overfitting and underfitting in our plot. Over fitting can be seen in 7th degree polynomial models and higher. In these models, the regression becomes too closely mapped to

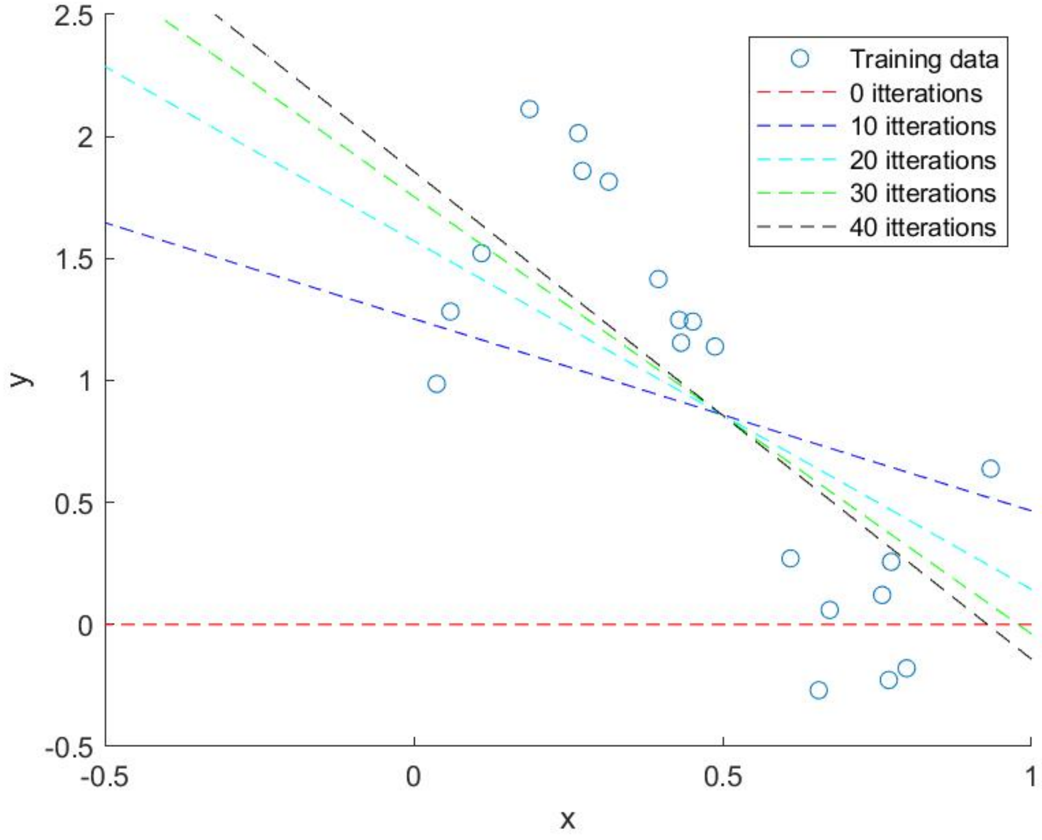

Figure 3: Linear Regressions from Incomplete Gradient Descnet

exactly the training data–and does not take the trend into account. As a result, our error spikes in the plot. Large coefficents are likely being used here to cater only to the training data coefficents, making these systems overfit to the data. We can also see a little bit of underfitting in the systems below a third degree polynomial. They have larger errors in the plot too as they are simply not capturing all of the phenomena that occur in our data set. As a result, these systems are underfit to the trianing (and test data).

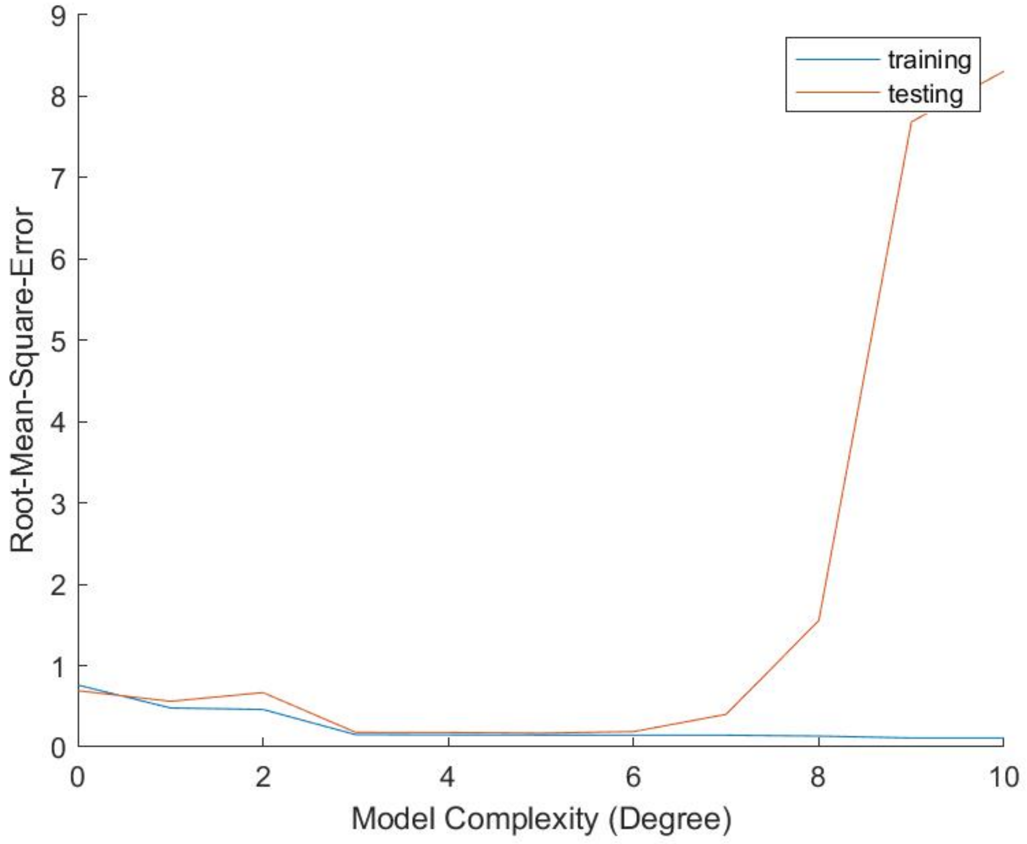

Figure 4: Root-Meas-Square Error as a function of model complexity